U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

2°ano – MIEIC – Abril 2017

Recolha de Lixo Inteligente

Conceção e Análise de Algoritmos

Turma 6 - Grupo C

Diogo Peixoto Pereira – up201504326
Gonçalo Vasconcelos Cunha Miranda Moreno – up201503871
Maria Eduarda Santos Cunha – up201506524

Índice

1. Introdução

O sistema atual de recolha de lixo consiste, tipicamente, na passagem de um camião de resíduos por todos os pontos de recolha espalhados numa cidade, recolhendo o conteúdo dos contentores e transportando-o para uma estação de tratamento de resíduos, que lida com os vários tipos adequadamente. Existem contentores específicos para cada tipo de resíduo, geralmente, amarelo para o plástico e embalagens, azul para papel e cartão, verde para vidro, vermelho para pilhas e, por fim, o preto para resíduos domésticos indiferenciados.

O nosso objetivo inicial com este trabalho era a simulação de um sistema de recolha mais avançado, que, ao invés de obrigar um camião a esta recolha de forma não prática e ineficaz, levasse a um percurso muito mais eficiente.

Nesta segunda fase do projeto, pretendemos agora verificar se existem contentores localizados em esquinas de cruzamentos entre ruas, sendo que estas ruas são escolhidas pelo utilizador. Consideramos que estas ruas têm nomes, o que nos permite explorar os algoritmos de Pesquisa Exata e Aproximada, já que estes lidam com a procura de *strings* em ficheiros de texto.

2. Formalização do Problema

A problematização a ser resolvida é identificar se existe ou não contentores em esquinas de cruzamentos entre ruas. Este problema acarreta consigo essencialmente 3 pequenas fases.

1. Descobrir se a rua que o utilizador procura existe e, nesse caso, qual o seu id.
2. Se existir, descobrir todos os vértices dessa rua e que são simultaneamente contentores.
3. Verificar se esses contentores formam esquinas.

2.1. Dados de Entrada

Ficheiros de texto com informação relativa a ruas e a nós, que traduzem um mapa, representado por um grafo, G = (V, E), cujos nós são aleatoriamente determinados como garagens, contentores ou aterros sanitários.

G – Grafo que abstrai mapa.

V – Vértices/nós que simbolizam as garagens, contentores e aterros sanitários.

Para os algoritmos de Pesquisa Exata e Aproximada, são necessárias as variáveis *string* com o nome da rua a procurar e *pattern length*.

2.2. Resultados Esperados

Dados os tipos de situações diferentes:

1. A rua existe:
   - Todos os ids correspondentes aos contentores que respeitam as condições de estarem nessa rua e formarem uma esquina;
   - Nenhum id, pois não existe simultaneamente nenhum vértice que represente um contentor, esteja nessa rua e forme uma esquina;
2. A rua não existe:
   - O programa termina, avisando que a rua não existe.

3. Solução

A solução pode ser dividida em três fases.

1. Através do algoritmo de Pesquisa Exata ou Aproximada, o utilizador insere o nome da rua que está à procura e verifica se existe.
Os algoritmos procuram um padrão semelhante entre a *string* inserida, correspondente ao nome da rua, e um ficheiro de texto com informação relativa ao id das ruas e seus respetivos nomes. Programamos os algoritmos para que, quando encontrassem a rua, nos devolvessem o seu id.

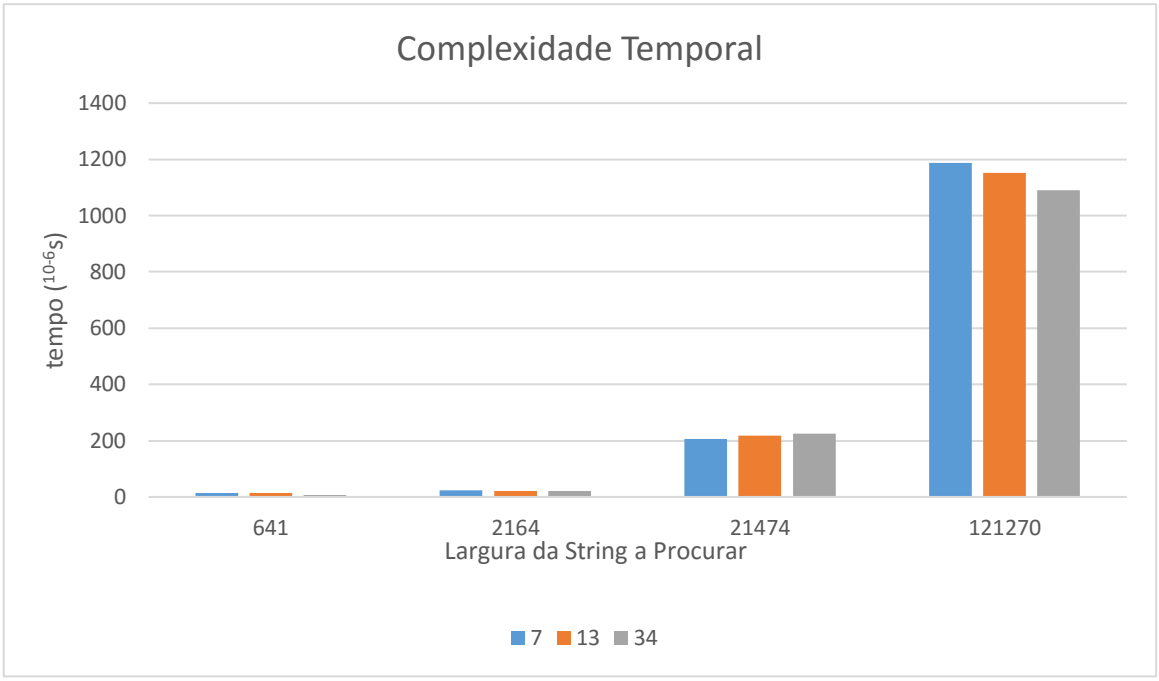
2. Percorrer todos os vértices, diretamente no grafo, correspondentes a contentores e com arestas correspondentes à rua pesquisa.

3. Verificar, para cada contentor selecionado, se possui arestas com mais ruas, de forma a perceber se forma uma esquina.

Quanto ao algoritmo de Pesquisa Exata:

A função auxiliar *kmp* tem dois parâmetros: o nome da rua e o tamanho do padrão. Pela análise do código, o algoritmo tem complexidade temporal O(n m), em que n é o nome da rua a procurar e m o tamanho do padrão a procurar.

A seguir, apresentamos um gráfico que mostra o tempo ocupado em µs testando vários tamanhos de *strings* a procurar (641, 2164, 21474 e 121270) e vários tamanhos de padrão (7, 13, 34).

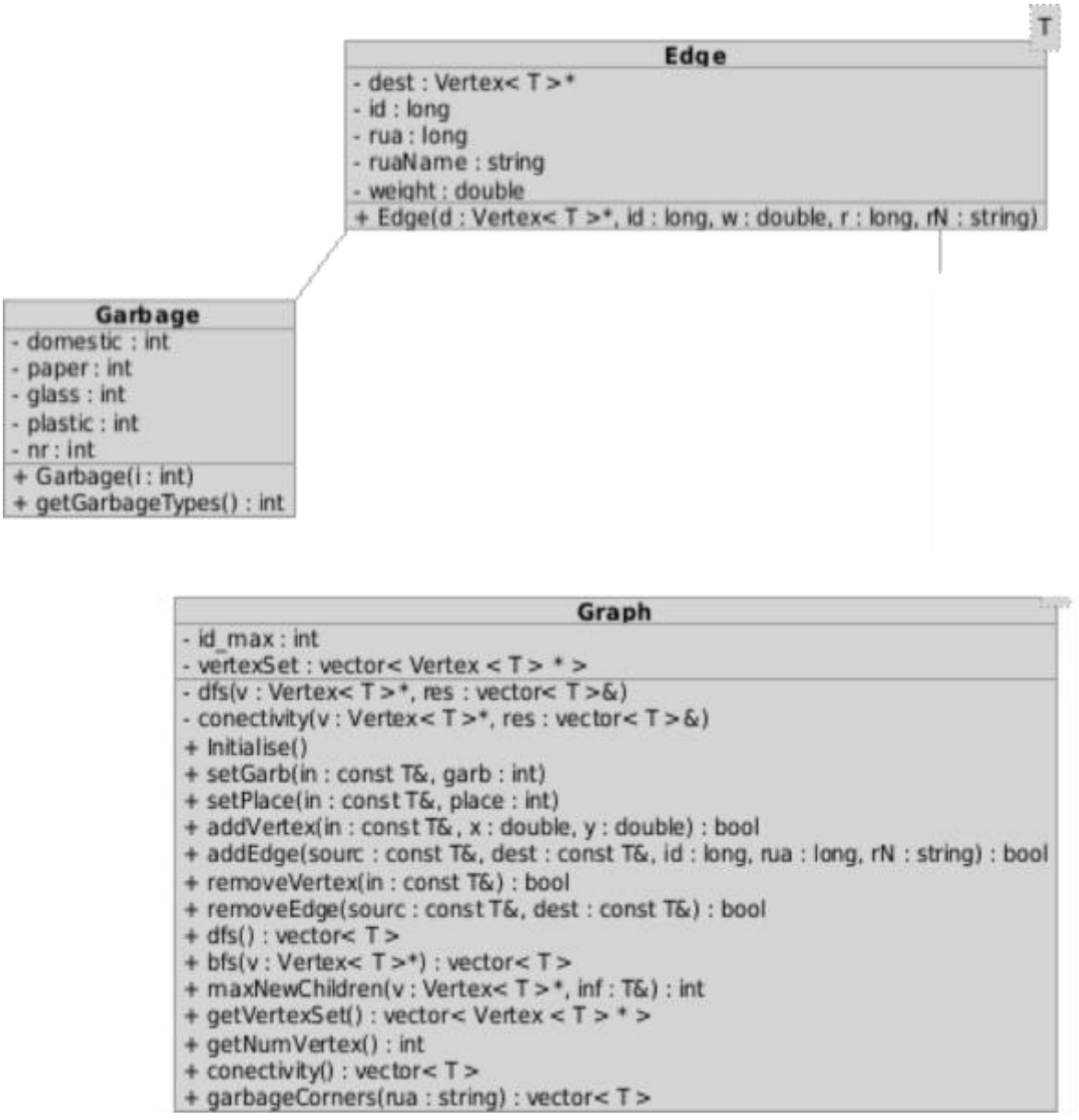


Nota:

Os resultados destes cálculos advieram de correr o algoritmo num sistema *Parrot OS* (*Linux* 64 bits), com um processador *i5-3360M* e compilado com g++, versão 6.3.0 20170415, com as *flags –Ofast*.

4. Diagrama de Classes

(Não se aplica verdadeiramente no conceito deste enunciado.)



5. Casos de Utilização

1. Saber se uma dada rua existe num mapa à escolha.
2. Descobrir todos os vértices presentes numa rua.
3. Confirmar a existência de contentores em esquinas, formadas com essa rua.

6. Dificuldades

A dificuldade mais relevante foi perceber como estruturar o problema. Não sabíamos se devíamos optar pela estratégia que acabamos por utilizar ou reduzir o grafo inicial para um grafo só com as esquinas e, daí, prosseguir com as pesquisas ou, ainda, dar ao utilizador a opção de inserir as duas ruas que deveriam formar uma esquina, testar se existiam e a formavam e, de seguida, verificar todos os vértices comuns às duas ruas e confirmar se eram contentores.

Ainda, trabalhar com a leitura de ficheiros acaba por se apresentar sempre como uma dificuldade, pois qualquer erro na escrita de ficheiros origina um erro no resultado final ao correr os algoritmos.

7. Contributos

Diogo Pereira - 33%

Gonçalo Moreno - 33%

Eduarda Cunha - 33%

8. Conclusão

Em relação ao primeiro enunciado do projeto, este segundo foi extremamente mais fácil, isto pois o caráter do problema em si já é muito mais simples do que o primeiro e, ainda, porque os algoritmos a implementar já estavam maioritariamente implementados nas aulas práticas.

Uma conclusão que pretendíamos retirar também, mas não foi possível, dada a falta de tempo, era a relação temporal entre os algoritmos de Pesquisa Exata e Aproximada. À partida, o tempo de execução do algoritmo de Pesquisa Exata seria inferior ao da Aproximada, mas dado que não a calculamos esta última não podemos concluir verdadeiramente nada. No entanto, o que esperávamos concluir era que se a discrepância de intervalos de tempo entre a implementação desses dois algoritmos fosse muito alta, tornava-se relevante a necessidade de um utilizador ideal (que escreve exatamente o nome da rua que pretende) para que utilizássemos apenas o algoritmo de Pesquisa Exata. Ao passo que, se essa discrepância não fosse significativa, talvez fosse mais vantajoso até possuir apenas o algoritmo de Pesquisa Aproximada, já que este conseguiria lidar com ambos os casos em que o utilizador é ideal ou não. Porém, se pusermos de lado as complexidades temporais e espaciais e tivermos apenas em conta o melhor resultado possível para o utilizador, o mais indicado seria executar primeiro o algoritmo de Pesquisa Exata e, se este falhasse, executar então o de Pesquisa Aproximada, garantindo que este não recebia informação desnecessária caso inserisse corretamente o nome da rua desde o início.

9. Bibliografia

[1] Narahari, Y. "8.4.2 Optimal Solution for TSP using Branch and Bound". Game Theory Lab. http://lcm.csa.iisc.ernet.in/dsa/node187.html (last accessed April 6, 2017)

[2] Stack Overflow. http://stackoverflow.com/questions/22985590/calculating-the-held-karp-lower-bound-for-the-traveling-salesmantsp

[3] MIT. "Branch and Bound". MIT OPEN COURSE WARE. https://ocw.mit.edu/courses/sloan-school-of-management/15-053-optimization-methods-in-management-science-spring-2013/tutorials/MIT15_053S13_tut10.pdf (last accessed April 4, 2017)

[4] "Travelling Salesman problem". Wikipedia. https://en.wikipedia.org/wiki/Travelling_salesman_problem#Related_problems

[5] Stack Overflow. http://stackoverflow.com/questions/22985590/calculating-the-held-karp-lower-bound-for-the-traveling-salesmantsp

[6] Kumar Singhal, Ritesh; Pandey, Dr. D. K. "Approximation of Shortest Path using Travelling Salesman Problem". International Journal of Advanced and Innovative Research. http://ijair.jctjournals.com/oct2012/t121015.pdf

[7] "Dijkstra's algorithm". Wikipedia. https://en.wikipedia.org/wiki/Dijkstra%27s_algorithmAGENTS

[8] Poole, David; Mackworth, Alan. "3.7.4 Branch and Bound". Artificial Intelligence. http://artint.info/html/ArtInt_63.html

[9] Gao, Jiyao. "Branch and bound (BB)". Northwestern Univeristy Process Optimization Open Textbook. https://optimization.mccormick.northwestern.edu/index.php/Branch_and_bound_(BB)

[10] Rossetti, R.; Rocha, A.P.; Camacho, R. 2016/2017. "Algoritmos em Strings". Moodle da Universidade do Porto (last accessed May 22, 2017)